Conceptos y Paradigmas de Lenguajes de Programación - 2023 - Segundo Parcial 1ra Fecha. T1
16/06/2023

Realice el parcial con lapicera, de otra forma se desaprobará el/los ejercicio/s.
Se considera presentismo cuando se realiza completamente un ejercicio.

Legajo: 18747/3						Corrigió:
Apellido y Nombres: Evequoz Guadalupe						
1	a B	b B				
2	a B-	b B-				
3	a B	b B				
4	a M	b M	c B	d B	e M	
					Resultado Final	APROBADO

Ejercicio 1 (30p)
Realice la pila de ejecución para el siguiente código:

a) por cadena estática
b) por cadena dinámica

```
Program Main;
Var x, y, z:integer;
a, b: array[1..6] of integer;

Procedure D(var x: integer; nombre y: integer);
  var h:integer;
  begin
     x:= f + 5;
     y:= y + 2;
     h:= x + 15;
  end;

Function F: integer;
  Var y:integer;
  Begin
    y:=1;
    b(x):= b(x) + 1;

    if(x < 6) then
       x:= x + 1;

    a(y):=a(y)+b(x)+3;
    a(x)=a(x) + 2;
    y:=y+3;
    return b(y);
  end

begin
    x:= 1;   y:= 1;
    for z:=1 to 6 do begin
            a(z):= abs(7-z);
            b(z):= z * 2;
    end;
    D(a(z), b(abs(6-x+y)));
    for z:=1 to 6 do  write (a(z), b(z));
end.
```

Nota: La forma de evaluación de este lenguaje es de izquierda a derecha. Abs es una función que retorna el valor absoluto de la operación recibida.

Ejercicio 2

a) (10pts) Clasifique las siguientes estructuras de datos de acuerdo a lo visto en la práctica.Justifique en cada caso:

i) **Java**

```
class Alumno {
  String nombre;
  String apellido;
  int edad;
  float promedio;
  String domicilio
```

Producto cartesiano

Realice el parcial con lapicera, de otra forma se desaprobará el/los ejercicio/s.
Se considera presentismo cuando se realiza completamente un ejercicio.

```
        public float getPromedio(){
            return this.promedio;
        }
    }
```

correspondence finite

ii) C

```
typedef struct _nodoArbol {
  void *info;
  struct _nodoArbol *hijoIzq;
  struct _nodoArbol *hijoDer;
} nodoArbol;

typedef struct _arbolBinario {
  int valor_guardado;
  nodoArbol *raiz
} arbolBinario;
```

producto cartesiano y recursión

producto cartesiano

b) (10 pts) Responda si las siguientes afirmaciones son V o F. Justifique en cada caso

i) Los lenguajes con sistema de tipos fuerte son siempre compilados V
ii) Las tuplas de python son un ejemplo de producto cartesiano F
iii) La unión y la unión discriminada no son seguras en ejecución F

Ejercicio 3

a) (15 pts) Dado el siguiente código en Java, establezca cuáles de las opciones indicadas más abajo son válidas como camino de ejecución. Justifique con una breve descripción del flujo de ejecución, caso contrario no se considerará válida la respuesta)

```
1  public class Java7MultiplesExceptions {
2
3      public static void main(String[] args) {
4          try{
5              for (int i = 1; i < 4; i++) {
6                  if(i==1){
7
8                      System.out.println(Integer.toString(i));
9                      rethrow("Primera");
10                 }
11                 else{
12                     if(i==2){
13                         System.out.println(Integer.toString(i));
14                         rethrow("Segunda");
15                     }
16                     else{
17                         if(i==3){
18                             System.out.println(Integer.toString(i));
19                             rethrow("Tercera");
20                         }
21                     }
22                 }
23             }
24         }catch(ThirdException e){
25             System.out.println(e.getMessage());
26     }
27
28 static void rethrow(String s) throws ThirdException {
29     try {
30         try {
31             if (s.equals("Primera")){
32                 throw new FirstException("Primera excepción");
33             }
34             else{
35                 if (s.equals("Segunda")){
36                     throw new SecondException("Segunda excepción");
37                 }
38                 else{
39                     throw new ThirdException("Tercera excepción");
40                 }
41             }
42         }catch (SecondException e) {
43             ThirdException e1=new ThirdException("Tercera excepción");
```

Realice el parcial con lapicera, de otra forma se desaprobará el/los ejercicio/s.
Se considera presentismo cuando se realiza completamente un ejercicio.

```
44                  throw e1;
45              }
46      }catch (FirstException e) {
47              ThirdException e1=new ThirdException("Tercera excepción");
48              throw e1;
49              }
50      }
51  }
```

i) Se imprime en pantalla "1" y luego "tercera excepción" y luego termina
ii) Se imprime en pantalla "1", "Primera excepción", "2", "segunda excepción", "3", "tercera excepción" y luego termina
iii) Se imprime en pantalla "1", "Tercera excepción", "2", "Tercera excepción", "3", "tercera excepción" y luego termina
iv) Ninguna de las anteriores

b) (10 pts) Indique si el resultado de intercambiar las líneas 4 y 5 (es decir, el try fuera del for) genera el mismo resultado de impresión. Justifique

Ejercicio 4

(25pts). Marcar si son verdaderas o falsas las siguientes afirmaciones. Acompañar la respuesta con una justificación, caso contrario, NO se tomarán cómo válidas

a. La sentencia for de ADA y Pascal son igualmente seguras (V) F ✗
b. El if de circuito corto puede prevenir errores en ejecución V (F) ✗
c. La sentencia yield de python equivale a hacer return V (F) ✓
d. En PL/1 si se genera una excepción, se ejecuta el manejador correspondiente y el control es pasado inmediatamente al programa principal V (F) ✓
e. La sentencia else de python en el manejo de excepciones se ejecuta solo si no se encontró ningún manejador asociado a la excepción en cuestión (V) F